```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TOD
OBJECT MODULE PLACED IN DATE.OBJ
COMPILER INVOKED BY:  PLM80 DATE.PLM PAGEWIDTH(100) DEBUG OPTIMIZE
```

```
                $title ('CP/M V3.0 Date and Time')
  1             tod:
                  do;

                /*
                  Revised:
                    14 Sept 81  by Thomas Rolander

                  Modifications:
                   Date: September 2,1982

                   Programmer: Thomas J. Mason

                   Changes:
                    The 'P' option was changed to the 'C'ontinuous option.
                    Also added is the 'S'et option to let the user set either
                    the time or the date.

                   Date: October 31,1982

                   Programmer: Bruce K. Skidmore

                   Changes:
                    Added Function 50 call to signal Time Set and Time Get.
                */

  2   1         declare PLM label public;

  3   1         mon1:
                 procedure (func,info) external;
  4   2            declare func byte;
  5   2            declare info address;
  6   2         end mon1;

  7   1         mon2:
                 procedure (func,info) byte external;
  8   2            declare func byte;
  9   2            declare info address;
 10   2         end mon2;

 11   1         mon2a:
                 procedure (func,info) address external;
 12   2            declare func byte;
 13   2            declare info address;
 14   2         end mon2a;

 15   1         declare xdos literally 'mon2a';

 16   1         declare fcb (1) byte external;
 17   1         declare fcb16 (1) byte external;
 18   1         declare tbuff (1) byte external;
```

CP/M Plus Ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
 19   1        RETURN$VERSION$FUNC:
                 procedure address;
 20   2           return MON2A(12,0);
 21   2         end RETURN$VERSION$FUNC;

 22   1        read$console:
                 procedure byte;
 23   2           return mon2 (1,0);
 24   2         end read$console;

 25   1        write$console:
                 procedure (char);
 26   2           declare char byte;
 27   2           call mon1 (2,char);
 28   2         end write$console;

 29   1        print$buffer:
                 procedure (buffer$address);
 30   2           declare buffer$address address;
 31   2           call mon1 (9,buffer$address);
 32   2         end print$buffer;

 33   1        READ$CONSOLE$BUFFER:
                 procedure (BUFF$ADR);
 34   2         declare BUFF$ADR address;
 35   2           call MON1(10,BUFF$ADR);
 36   2         end READ$CONSOLE$BUFFER;

 37   1        check$console$status:
                 procedure byte;
 38   2           return mon2 (11,0);
 39   2         end check$console$status;


 40   1        terminate:
                 procedure;
 41   2           call mon1 (0,0);
 42   2         end terminate;

 43   1        crlf:
                 procedure;
 44   2           call write$console (0dh);
 45   2           call write$console (0ah);
 46   2         end crlf;

            /*************************************************

                    Time & Date ASCII Conversion Code

             *************************************************/
 47   1      declare BUFFER$ADR structure (
                    MAX$CHARS byte,
                    NUMB$OF$CHARS byte,
                    CONSOLE$BUFFER(21) byte)
                    initial(21,0,0,0,0,0,0,0,0,0,0,0,
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        0,0,0,0,0,0,0,0,0,0,0);

 48  1      declare tod$adr address;
 49  1      declare tod based tod$adr structure (
              opcode byte,
              date address,
              hrs byte,
              min byte,
              sec byte,
              ASCII (21) byte );

 50  1      declare string$adr address;
 51  1      declare string based string$adr (1) byte;
 52  1      declare index byte;

 53  1      declare lit literally 'literally',
               forever lit 'while 1',
               word lit 'address';

              /* - - - - - - - - - - - - - - - - - - - - - - - - */
 54  1          emitchar:
                  procedure(c);
 55  2              declare c byte;
 56  2                 string(index := index + 1) = c;
 57  2              end emitchar;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/
 58  1          emitn:
                  procedure(a);
 59  2              declare a address;
 60  2              declare c based a byte;
 61  2                 do while c <> '$';
 62  3                    string(index := index + 1) = c;
 63  3                    a = a + 1;
 64  3                 end;
 65  2              end emitn;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 66  1          emit$bcd:
                  procedure(b);
 67  2              declare b byte;
 68  2                 call emitchar('0'+b);
 69  2              end emit$bcd;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 70  1          emit$bcd$pair:
                  procedure(b);
 71  2              declare b byte;
 72  2                 call emit$bcd(shr(b,4));
 73  2                 call emit$bcd(b and 0fh);
 74  2              end emit$bcd$pair;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 75  1          emit$colon:
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                  procedure(b);
 76   2           declare b byte;
 77   2              call emit$bcd$pair(b);
 78   2              call emitchar(':');
 79   2           end emit$colon;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 80   1           emit$bin$pair:
                  procedure(b);
 81   2           declare b byte;
 82   2              call emit$bcd(b/10);
 83   2              call emit$bcd(b mod 10);
 84   2           end emit$bin$pair;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 85   1           emit$slant:
                  procedure(b);
 86   2           declare b byte;
 87   2              call emit$bin$pair(b);
 88   2              call emitchar('/');
 89   2           end emit$slant;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 90   1           declare chr byte;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/

 91   1           gnc:
                  procedure;
                  /* get next command byte */
 92   2              if chr = 0 then return;
 94   2              if index = 20 then
 95   2              do;
 96   3                 chr = 0;
 97   3                 return;
 98   3              end;
 99   2              chr = string(index := index + 1);
100   2           end gnc;

              /*- - - - - - - - - - - - - - - - - - - - - - - - -*/
101   1           deblank:
                  procedure;
102   2              do while chr = ' ';
103   3                 call gnc;
104   3              end;
105   2           end deblank;

106   1           numeric:
                  procedure byte;
                  /* test for numeric */
107   2              return (chr - '0') < 10;
108   2           end numeric;

109   1           scan$numeric:
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                  procedure(lb,ub) byte;
110   2           declare (lb,ub) byte;
111   2           declare b byte;
112   2              b = 0;
113   2              call deblank;
114   2              if not numeric then go to error;
116   2              do while numeric;
117   3                 if (b and 1110$0000b) <> 0 then go to error;
119   3                 b = shl(b,3) + shl(b,1); /* b = b * 10 */
120   3                 if carry then go to error;
122   3                 b = b + (chr - '0');
123   3                 if carry then go to error;
125   3                 call gnc;
126   3              end;
127   2              if (b < lb) or (b > ub) then go to error;
129   2              return b;
130   2           end scan$numeric;

131   1           scan$delimiter:
                  procedure(d,lb,ub) byte;
132   2           declare (d,lb,ub) byte;
133   2              call deblank;
134   2              if chr <> d then go to error;
136   2              call gnc;
137   2              return scan$numeric(lb,ub);
138   2           end scan$delimiter;

139   1     declare base$year lit '78',   /* base year for computations */
                    base$day  lit '0',    /* starting day for base$year 0..6 */
                    month$size (*) byte data
                    /* jan feb mar apr may jun jul aug sep oct nov dec */
                    (   31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31),
                    month$days (*) word data
                    /* jan feb mar apr may jun jul aug sep oct nov dec */
                    (  000,031,059,090,120,151,181,212,243,273,304,334);

140   1           leap$days:
                  procedure(y,m) byte;
141   2           declare (y,m) byte;
                  /* compute days accumulated by leap years */
142   2           declare yp byte;
143   2              yp = shr(y,2); /* yp = y/4 */
144   2              if (y and 11b) = 0 and month$days(m) < 59 then
                       /* y not 00, y mod 4 = 0, before march, so not leap yr */
145   2                 return yp - 1;
                  /* otherwise, yp is the number of accumulated leap days */
146   2              return yp;
147   2           end leap$days;

148   1           declare word$value word;

149   1           get$next$digit:
                  procedure byte;
                  /* get next lsd from word$value */
150   2           declare lsd byte;
151   2              lsd = word$value mod 10;
152   2              word$value = word$value / 10;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
153  2          return lsd;
154  2        end get$next$digit;

155  1        bcd:
              procedure (val) byte;
156  2        declare val byte;
157  2          return shl((val/10),4) + val mod 10;
158  2        end bcd;

159  1        declare (month, day, year, hrs, min, sec) byte;

160  1        set$date:
              procedure;
161  2        declare (i, leap$flag) byte; /* temporaries */
162  2          month = scan$numeric(1,12) - 1;
             /* may be feb 29 */
163  2          if (leap$flag := month = 1) then i = 29;
165  2           else i = month$size(month);
166  2          day   = scan$delimiter('/',1,i);
167  2          year  = scan$delimiter('/',base$year,99);
             /* ensure that feb 29 is in a leap year */
168  2          if leap$flag and day = 29 and (year and 11b) <> 0 then
169  2        /* feb 29 of non-leap year */ go to error;
             /* compute total days */
170  2          tod.date = month$days(month)
                        + 365 * (year - base$year)
                        + day
                        - leap$days(base$year,0)
                        + leap$days(year,month);

171  2         end SET$DATE;

172  1        SET$TIME:
              procedure;
173  2          tod.hrs   = bcd (scan$numeric(0,23));
174  2          tod.min   = bcd (scan$delimiter(':',0,59));
175  2          if tod.opcode = 2
               then
             /* date, hours and minutes only */
176  2           do;
177  3             if chr = ':'
                   then i = scan$delimiter (':',0,59);
179  3             tod.sec = 0;
180  3           end;
             /* include seconds */
181  2           else tod.sec   = bcd (scan$delimiter(':',0,59));
182  2        end set$time;

183  1        bcd$pair:
              procedure(a,b) byte;
184  2        declare (a,b) byte;
185  2          return shl(a,4) or b;
186  2        end bcd$pair;


187  1        compute$year:
              procedure;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                /* compute year from number of days in word$value */
188   2         declare year$length word;
189   2           year = base$year;
190   2              do forever;
191   3                 year$length = 365;
192   3                 if (year and 11b) = 0 then /* leap year */
193   3                    year$length = 366;
194   3                 if word$value <= year$length then
195   3                    return;
196   3                 word$value = word$value - year$length;
197   3                 year = year + 1;
198   3              end;
199   2         end compute$year;

200   1       declare week$day  byte, /* day of week 0 ... 6 */
                      day$list (*) byte data ('Sun$Mon$Tue$Wed$Thu$Fri$Sat$'),
                      leap$bias byte; /* bias for feb 29 */

201   1          compute$month:
                   procedure;
202   2             month = 12;
203   2             do while month > 0;
204   3                if (month := month - 1) < 2 then /* jan or feb */
205   3                   leapbias = 0;
206   3                if month$days(month) + leap$bias < word$value then return;
208   3             end;
209   2          end compute$month;

210   1       declare date$test byte,    /* true if testing date */
                      test$value word;   /* sequential date value under test */

211   1          get$date$time:
                   procedure;
                   /* get date and time */
212   2             hrs = tod.hrs;
213   2             min = tod.min;
214   2             sec = tod.sec;
215   2             word$value = tod.date;
                   /* word$value contains total number of days */
216   2             week$day = (word$value + base$day - 1) mod 7;
217   2             call compute$year;
                   /* year has been set, word$value is remainder */
218   2             leap$bias = 0;
219   2             if (year and 11b) = 0 and word$value > 59 then
220   2                /* after feb 29 on leap year */ leap$bias = 1;
221   2             call compute$month;
222   2             day = word$value - (month$days(month) + leap$bias);
223   2             month = month + 1;
224   2          end get$date$time;

225   1          emit$date$time:
                   procedure;
226   2             call emitn(.day$list(shl(week$day,2)));
227   2             call emitchar(' ');
228   2             call emit$slant(month);
229   2             call emit$slant(day);
230   2             call emit$bin$pair(year);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
231   2           call emitchar(' ');
232   2           call emit$colon(hrs);
233   2           call emit$colon(min);
234   2           call emit$bcd$pair(sec);
235   2         end emit$date$time;

236   1         tod$ASCII:
                  procedure (parameter);
237   2          declare parameter address;
238   2          declare ret address;
239   2             ret = 0;
240   2             tod$adr = parameter;
241   2             string$adr = .tod.ASCII;
242   2             if tod.opcode = 0 then
243   2             do;
244   3                call get$date$time;
245   3                index = -1;
246   3                call emit$date$time;
247   3             end;
                     else
248   2             do;
249   3                if (tod.opcode = 1) or
                        (tod.opcode = 2) then
250   3                do;
251   4                   chr = string(index:=0);
252   4                   call set$date;
253   4                   call set$time;
254   4                   ret = .string(index);
255   4                end;
                        else
256   3                do;
257   4                   go to error;
258   4                end;
259   3             end;
260   2         end tod$ASCII;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
        /*************************************************************
         *************************************************************/


261   1         declare lcltod structure (
                    opcode byte,
                    date address,
                    hrs byte,
                    min byte,
                    sec byte,
                    ASCII (21) byte );

262   1         declare datapgadr address;
263   1         declare datapg based datapgadr address;

264   1         declare extrnl$todadr address;
265   1         declare extrnl$tod based extrnl$todadr structure (
                    date address,
                    hrs byte,
                    min byte,
                    sec byte );
```

```
266  1        declare i byte;
267  1        declare ret address;

268  1        display$tod:
              procedure;
269  2           lcltod.opcode = 0; /* read tod */
270  2           call mon1(50,.(26,0,0,0,0,0,0,0)); /* BIOS TIME GET SIGNAL */
271  2           call move (5,.extrn1$tod.date,.lcltod.date);
272  2           call tod$ASCII (.lcltod);
273  2           call write$console (0dh);
274  2           do i = 0 to 20;
275  3              call write$console (lcltod.ASCII(i));
276  3           end;
277  2        end display$tod;

278  1        comp:
              procedure (cnt,parmadr1,parmadr2) byte;
279  2        declare (i,cnt) byte;
280  2        declare (parmadr1,parmadr2) address;
281  2        declare parm1 based parmadr1 (5) byte;
282  2        declare parm2 based parmadr2 (5) byte;
283  2           do i = 0 to cnt-1;
284  3              if parm1(i) <> parm2(i)
                    then return 0;
286  3           end;
287  2           return 0ffh;
288  2        end comp;


             /**************************************

              Main Program

             **************************************/

289  1        declare last$dseg$byte byte initial (0);
290  1        declare CURRENT$VERSION address initial (0);
291  1        declare CPM30 byte initial (030h);
292  1        declare MPM byte initial (01h);

293  1      PLM:
         do;
294  2        CURRENT$VERSION = RETURN$VERSION$FUNC;
295  2        if (low(CURRENT$VERSION) >= CPM30) and (high(CURRENT$VERSION) <> MPM) then
296  2        do;
297  3           datapgadr = xdos (49,.(03ah,0));
298  3           extrn1$todadr = xdos(49,.(03ah,0)) + 58H;
299  3           if (FCB(1) = 'C') then
300  3           do while FCB(1) = 'C';
301  4              if comp(5,.extrn1$tod.date,.lcltod.date) = 0 then
302  4                 call display$tod;
303  4              if check$console$status then
304  4              do;
305  5                 ret = read$console;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
306  5              fcb(1) = 0;
307  5            end;
308  4          end;
               else
309  3            if (FCB(1) = ' ') then
310  3            do;
311  4               call display$tod;
312  4            end;
                  else
313  3               if (FCB(1) = 'S')
                     then do;
315  4                   call crlf;
316  4                   call print$buffer(.('Enter today''s date (MM/DD/YY): ','$'));
317  4                   call move(21,.(0000000000000000000000),.buffer$adr,console$buffer);
318  4                   call read$console$buffer(.buffer$adr);
319  4                   if buffer$adr.numb$of$chars > 0
                         then do;
321  5                      call move(21,.buffer$adr.console$buffer,.lcltod,ASCII);
322  5                      tod$adr = .lcltod;
323  5                      string$adr = .tod,ASCII;
324  5                      chr = string(index := 0);
325  5                      call set$date;
326  5                      call move(2,.lcltod.date,.extrnl$tod.date);
327  5                   end;  /* date initialization */
328  4                   call crlf;
329  4                   call print$buffer(.('Enter the time (HH:MM:SS):     ','$'));
330  4                   call move(21,.(0000000000000000000000),.buffer$adr,console$buffer);
331  4                   call read$console$buffer(.buffer$adr);
332  4                   if buffer$adr.numb$of$chars > 0
                          then do;
334  5                      call move(21,.buffer$adr.console$buffer,.lcltod,ASCII);
335  5                      tod$adr = .lcltod;
336  5                      string$adr = .tod,ASCII;
337  5                      chr = string(index := 0);
338  5                      call set$time;
339  5                      call crlf;
340  5                      call print$buffer(.('Press any key to set time ','$'));
341  5                      ret = read$console;
342  5                      call move(3,.lcltod.hrs,.extrnl$tod.hrs);
343  5                      call mon1(50,.(26,0,0ffh,0,0,0,0,0,0)); /* BIOS TIME SET SIGNAL */
344  5                    end;
345  4                    call crlf;
346  4                 end;
347  3                else do;
348  4                   call move (21,.tbuff(1),.lcltod,ASCII);
349  4                   lcltod.opcode = 1;
350  4                   call tod$ASCII (.lcltod);
351  4                   call crlf;
352  4                   call print$buffer (.('Strike key to set time','$'));
353  4                   ret = read$console;
354  4                   call move (5,.lcltod.date,.extrnl$tod.date);
355  4                   call mon1(50,.(26,0,0ffh,0,0,0,0,0,0)); /* BIOS TIME SET SIGNAL */
356  4                   call crlf;
357  4                 end;
358  3          call terminate;
359  3          end;
               else
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 360  2          do;
 361  3            call CRLF;
 362  3            call PRINT$BUFFER(.('ERROR: Requires CP/M3.','$'));
 363  3            call CRLF;
 364  3            call TERMINATE;
 365  3          end;
 366  2        end;

 367  1        error:
                do;
 368  2          call crlf;
 369  2          call print$buffer (.('ERROR: Illegal time/date specification.','$'));
 370  2          call terminate;
 371  2        end;

 372  1      end tod;
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 090DH   2317D
    VARIABLE AREA SIZE = 0079H    121D
    MAXIMUM STACK SIZE = 000EH     14D
    580 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
0000 DATE
0000 MCD80A
0004 BDISK     0080 BUFF       0080 BUFFA      0050 CMDRV      0000 CPU
007C CR                006D DOLLA      005C FCB        006C FCB16      005C FCBA
005C IFCB      005C IFCBA      0003 IOBYTE     0053 LEN0       0056 LEN1
0006 MAXB      0006 MEMSIZ     0005 MON1       0005 MON2       0005 MON2A
0005 MON3      0000 OFFSET     006E PARMA      0051 PASS0      0054 PASS1
007F RO                007D RR         007D RRECA      007F RRECO      005C SFCB
0080 TBUFF
0000 TOD
0BD8 MEMORY    0295 PLM        0493 RETURNVERSIONFUNC
049C READCONSOLE                04A5 WRITECONSOLE               0B5F CHAR
04B5 PRINTBUFFER                0B60 BUFFERADDRESS
04C5 READCONSOLEBUFFER          0B62 BUFFADR    04D5 CHECKCONSOLESTATUS
04DE TERMINATE 04E7 CRLF        0B64 BUFFERADR  0B7B TODADR     0B7D STRINGADR
0B7F INDEX     04F2 EMITCHAR   0B80 C          0509 EMITN      0B81 A
0538 EMITBCD   0B83 B          0546 EMITBCDPAIR                0B84 B
0561 EMITCOLON 0B85 B          0572 EMITBINPAIR                0B86 B
0597 EMITSLANT 0B87 B          0B88 CHR        05A8 GNC        05D2 DEBLANK
05E1 NUMERIC   05EA SCANNUMERIC                 0B89 LB         0B8A UB
0B8B B         0665 SCANDELIMITER               0B8C D          0B8D LB
0B8E UB                0180 MONTHSIZE  018C MONTHDAYS  068F LEAPDAYS   0B8F Y
0B90 M         0B91 YP         0B92 WORDVALUE  06CB GETNEXTDIGIT
0B94 LSD       06E9 BCD        0B95 VAL        0B96 MONTH      0B97 DAY
0B98 YEAR      0B99 HRS        0B9A MIN        0B9B SEC        070C SETDATE
0B9C I         0B9D LEAPFLAG   07C4 SETTIME    082E BCDPAIR    0B9E A
0B9F B         0840 COMPUTEYEAR                0BA0 YEARLENGTH 0BA2 WEEKDAY
01A4 DAYLIST   0BA3 LEAPBIAS   087E COMPUTEMONTH               0BA4 DATETEST
0BA5 TESTVALUE 08BC GETDATETIME                 093A EMITDATETIME
0980 TODASCII  0BA7 PARAMETER  0BA9 RET        0BAB LCLTOD     0BC6 DATAPGADR
0BC8 EXTRNLTODADR               0BCA I          0BCB RET        09ED DISPLAYTOD
0A43 COMP      0BCD CNT        0BCE PARMADR1   0BD0 PARMADR2   0BD2 I
0BD3 LASTDSEGBYTE               0BD4 CURRENTVERSION             0BD6 CPM30
0BD7 MPM       0482 ERROR
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
0000 DATE#
0000 MCD80A#
0000 TOD#
0493 19    0493 20    049C 21    049C 22    049C 23
04A5 24    04A5 25    04A9 27    04B4 28    04B5 29
04BB 31    04C4 32    04C5 33    04CB 35    04D4 36
04D5 37    04D5 38    04DE 39    04DE 40    04DE 41
04E6 42    04E7 43    04E7 44    04EC 45    04F1 46
04F2 54    04F6 56    0508 57    0509 58    050F 61
0518 62    052D 63    0534 64    0537 65    0538 66
053C 68    0545 69    0546 70    054A 72    0557 73
0560 74    0561 75    0565 77    056C 78    0571 79
0572 80    0576 82    0586 83    0596 84    0597 85
059B 87    05A2 88    05A7 89    05A8 91    05A8 92
05B0 93    05B1 94    05B9 96    05BE 97    05BF 98
05BF 99    05D1 100   05D2 101   05D2 102   05DA 103
05DD 104   05E0 105   05E1 106   05E1 107   05EA 108
05EA 109   05F0 112   05F5 113   05F8 114   05FF 115
0602 116   0609 117   0613 118   0616 119   0627 120
062C 121   062F 122   0639 123   063E 124   0641 125
0644 126   0647 127   065E 128   0661 129   0665 130
0665 131   0670 133   0673 134   067D 135   0680 136
0683 137   068F 138   068F 140   0695 143   069F 144
06C2 145   06C7 146   06CB 147   06CB 149   06CB 151
06DA 152   06E5 153   06E9 154   06E9 155   06ED 157
070C 158   070C 160   070C 162   0717 163   0726 164
072E 165   073B 166   074A 167   0757 168   0775 169
0778 170   07C3 171   07C4 172   07C4 173   07D7 174
07ED 175   07F6 177   07FE 178   080B 179   0814 180
0817 181   082D 182   082E 183   0834 185   0840 186
0840 187   0840 189   0845 190   0845 191   084B 192
0855 193   085B 194   0867 195   086B 196   0876 197
087A 198   087D 199   087E 201   087E 202   0883 203
088C 204   0898 205   089D 206   08B7 207   08B8 208
08BB 209   08BC 211   08BC 212   08C7 213   08D0 214
08D9 215   08E4 216   08F1 217   08F4 218   08F9 219
0914 220   0919 221   091C 222   0937 223   0939 224
093A 225   093A 226   094B 227   0950 228   0957 229
095E 230   0965 231   096A 232   0971 233   0978 234
097F 235   0980 236   0986 239   098C 240   0992 241
0999 242   09A2 244   09A5 245   09AA 246   09AD 247
09B0 249   09C7 251   09D3 252   09D6 253   09D9 254
09E6 255   09E9 257   09EC 258   09EC 259   09EC 260
09ED 268   09ED 269   09F2 270   09FA 271   0A0E 272
0A14 273   0A19 274   0A27 275   0A38 276   0A42 277
0A43 278   0A52 283   0A62 284   0A7D 285   0A80 286
0A8A 287   0A8D 288   0292 293   0298 294   029E 295
02BB 297   02C6 298   02D5 299   02DD 300   02E5 301
02F8 302   02FB 303   0302 305   030B 306   0310 307
0310 308   0316 309   031E 311   0321 312   0324 313
032C 315   032F 316   0335 317   0345 318   034B 319
0354 321   0364 322   036A 323   0374 324   0380 325
0383 326   0396 327   0396 328   0399 329   039F 330
03AF 331   03B5 332   03BE 334   03CE 335   03D4 336
03DE 337   03EA 338   03ED 339   03F0 340   03F6 341
03FF 342   0414 343   041C 344   041C 345   041F 346
0422 348   0432 349   0437 350   043D 351   0440 352
0446 353   044F 354   0462 355   046A 356   046D 357
046D 358   0470 359   0473 361   0476 362   047C 363
047F 364   0482 365   0482 367   0485 368   0488 369
048E 370   0491 372
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____